A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT

A Semantic Web Approach to Simplifying Trigger-Action Programming in the IoT

Fulvio Corno, Luigi De Russis, Alberto Monge Roffarello, Politecnico di Torino, Italy

Programming environments for End-User Development in the Internet of Things (IoT) allow end-users to customize the joint behavior of their IoT objects, typically through trigger-action rules. Unfortunately, they mainly adopt highly technology-dependent models, i.e., they often categorize devices and services by manufacturers or brands. An example of how such low-level of abstraction could be overcome is EUPont, an ontological model to provide an abstract and technology-independent representation for IoT end-user programming environments. In this paper, we show how EUPont could improve the rule composition process, and we evaluate its expressiveness by presenting an automated translation procedure of 290,963 rules shared on the IFTTT programming environment. Results show that EUPont improves the rule composition process for end-users in terms of correctness and needed time. Furthermore, the evaluation confirms that the model is more expressive than the one adopted by IFTTT, one of the most common solutions in this field.

Keywords: End-User Development, Internet of Things, Semantic Web, Trigger-Action Programming, Ontology

In the emergent Internet of Things (IoT) world, end-users with and without programming skills are willing to customize the joint behavior of IoT objects based on their personal needs. In this context, programming environments for End User Development (EUD) such as IFTTT [1] are becoming increasingly common. They allow the definition of simple *IoT applications*, i.e., connections between pairs of IoT devices and services. Such applications are typically expressed as trigger-action rules, where an action is automatically executed when an event (the trigger) is detected. Unfortunately, contemporary programming environments adopt highly technology-dependent representation models, i.e., they categorize devices and services by manufacturers or brands. With this approach, end-users have to define several similar rules to satisfy their needs, even if the defined rules perform the same logical operation. Furthermore, end-users must be aware of every single object they may encounter before creating their rules, without the possibility to define their applications for yet unknown places, users, or IoT devices and services. For instance, the rule "*if* the bedroom motion sensor detects a movement, *then* turn the table lamps in the bedroom on" refers to a specific motion sensor and to a specific lamp, and it cannot be generalized to reproduce the same behavior (i.e., turning the lights on) in the kitchen. With this lack of discovery and adaptation features, the expressive power of the definable rules is very poor.

An example of how such "low-level" of abstraction could be overcome is provided by EUPont [2], a "high-level" ontological model we defined to provide an abstract and technology-independent representation for IoT end-user programming environments. With such a model, a user interested in controlling her lights at home could simply define one single rule, e.g., "*if* I enter a room at home, *then* turn the lights in the room on", independently from the lights manufacturer and from the involved location.
In this paper, we show how EUPont could improve the rule composition process by describing a user

study we conducted [2], and we discuss and evaluate the expressive power of the model. To this purpose, we first highlight the characteristics of the trigger-action programming approach adopted in EUPont. Then, we show an automated translation procedure that allowed us to represent in EUPont 290,963 rules publicly shared on IFTTT [3], one of the most common end-user programming environments for the IoT. The adopted trigger-action approach, along with the translation results, shows that EUPont is more expressive than the representation adopted by IFTTT, and it is fully compatible with rules already defined in a low-level representation. In addition, the translation confirms that EUPont significantly reduces the number of rules needed to satisfy all users' needs.

End-User Development in the IoT

As defined by Lieberman et al. [4], EUD is "a set of methods, techniques, and tools that allow users of software systems, who are acting as non-professional software developers, at some point to create, modify or extend a software artifact". Starting from iCAP [5], a visual rule-based system for PC to create context-aware applications, EUD is becoming a promising approach for the IoT. Several interfaces and tools for end-user programming in the IoT are present both in the literature (e.g., [6, 7]) and as off-the-shelf products (e.g., IFTTT [1], Atooma [8]). Such applications allow the definition of rules, typically in the trigger-action form, without needing to write any code. The trigger-action format is one of the most used in EUD, and can be useful and usable for end-user programming in IoT settings like smart homes [9, 10], and for cross-devices usage [11].

Among the commercial tools, IFTTT is widely used and appreciated. IFTTT is a success in terms of user understandability and ease of use, with more than 1 million rules created by its users [12]. It allows the composition of simple connections (named *applets*) between more than 400 supported IoT objects (named *services*) [13]. The supported objects range from commercial devices (e.g, the Nest thermostat), to web or mobile services (e.g., Facebook). Applets, at least in the free version, can include a single trigger and a unique action, and can be composed by using a wizard-based procedure (a screenshot is shown in Figure 1).

Although EUD in the IoT has recently gained interest, interoperability and scalability challenges remain. Each of the 400 devices and services modeled by IFTTT, in fact, has its own different properties, with associated various triggers and actions. With the spread of new smart "things", the amount of information may become too high and cluttered. Furthermore, recent studies criticize the expressive power of IFTTT since it is rather poor in terms of trigger and action composition [9, 14], while others [15, 9] show that the trigger-action approach should be further investigated to cope with the evolving IoT world. From this evidence, some works have tried to improve models and interfaces for EUD in the IoT, e.g., by allowing users to define context-dependent applications [16]. In agreement with the work of Ur *et al.* [9], we claim that that, besides considering the context, a new breed of programming environments should be designed to support a "higher level" representation of IoT devices and services. For this purpose, we developed EUPont [2], an ontology that considers the entities of the IoT ecosystem based on their categories and capabilities, and allows the definition of abstract trigger-action rules that can be automatically adapted to different contextual situations. EUPont is described in the next Sections, along with its evaluation in terms of usefulness and expressiveness.

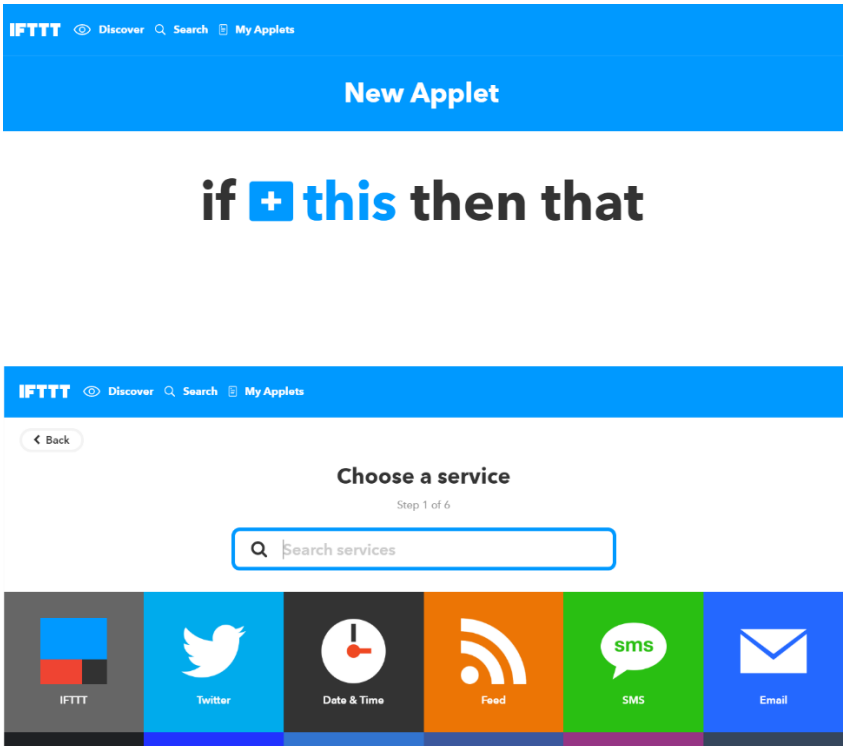

Figure 1: Applet Creation in IFTTT

EUPont: Programming the IoT by Functionality

To better explain our idea of EUD in the IoT we report a simple scenario:

Susan is an architect that likes bright environments, especially when she is working. She is also interested in new technologies, and she equipped her home with intelligent lights and movement sensors. She installed some Philips Hue lamps in the kitchen, and a Stack Lighting lamp in the living room, while all the rooms are surveilled by a Nest Cam. Her office, instead, is equipped with LIFX lights, and with a Samsung SmartThings hub that controls all the doors.

Suppose that Susan would like to automatically increase the environment brightness when she enters a specific closed place, e.g., by turning the lights on. By using IFTTT, she should define the following rules:

- *If* the kitchen's Nest Cam recognizes me, *then* turn the Philips Hue located in the kitchen on.
- *If* the living room's Nest Cam recognizes me, *then* turn the Stack Lighting lamp located in the living room on.
- *If* the SmartThings hub detects that the office's door opens, *then* turn the LIFX lights located in the office on.

In other words, Susan should define at least 3 different rules even if they perform the same logical operation, and she must be aware of all the technologies and branded-products involved, by specifying the right object for each rule. Furthermore, even with an authorization, if Susan enters a place different than her home or office, the rules that she already defined cannot be used to control the new environment.

We would like to help Susan in setting up her environments with one rule, only:

- *If* I enter a closed space, *then* turn its lights on

Such a rule is abstract and technology/brand independent: it does not require manually specifying the involved objects and places, and it can be used for different contextual situations (e.g., for the kitchen, the living room, and the office). In other words, the trigger and the action of the rule define the behavior in which Susan is interested, i.e., the **functionality** that Susan would like to obtain from the involved IoT objects.

A Semantic Model

To allow the definition of abstract and technology/brand independent rules, we developed EUPont, a high-level representation for End User Development in the IoT.

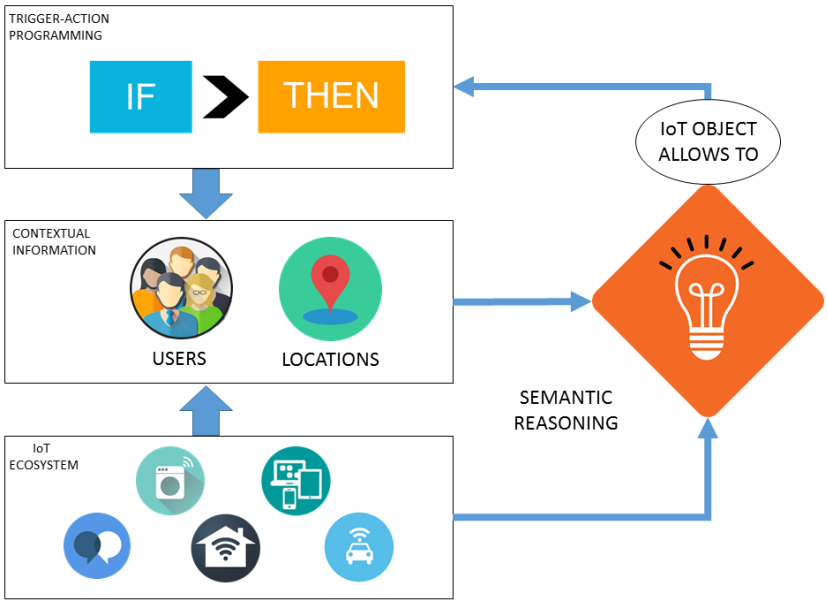

Figure 2: The EUPont Ontology Structure

EUPont models the objects that compose the IoT ecosystem based on their categories (e.g., lighting systems, user devices, smart appliances, social networks) and their capabilities (e.g., sensing, actuating, communication). EUPont is built as an ontology (available at http://elite.polito.it/ontologies/eupont.owl), thus offering all the advantages of the Semantic Web framework. Semantic Web technologies shall support the seamless integration of data, on-the-fly composition and interoperation of Web services [17]. With a semantic representation, the model can easily answer queries such as "which IoT devices or services can perform a particular action?" or "which IoT devices or services can generate a particular event?".

Figure 2 shows the overall structure of EUPont. The ontology is composed of four main blocks: *Trigger-Action Programming*, *Contextual Information*, *IoT Ecosystem*, and *Semantic Reasoning*.

Trigger-Action Programming

The **Trigger-Action Programming** block allows the definition of abstract and technology/brand independent trigger-action rules. Triggers and actions are organized in a hierarchy, to allow the choice among different levels of abstraction. Depending on the granularity of user needs, EUPont defines two

main levels of abstraction, that we call *Medium Level* and *High Level*. Figure 4 shows the hierarchy for some lighting-related actions. For instance, the action defined by Susan (i.e., “turn the lights on”) is included in the “illuminate the place” action, along with, for example, “open the blinds”.

EUPont rules may support multiple triggers and actions. Triggers can be chained by means of the OR logical operator, while actions can be chained with the AND operator. Furthermore, for each trigger and action, users can specify up to three additional restrictions, namely *Who, Where,* and *What.* Similar restrictions are defined in the work of Desolda et al. [17]. The restriction *What* can be used to specify a list of details to associate with the trigger or the action, e.g., a temperature threshold to monitor, or the text of a message to be sent. Instead, *Who* defines the involved user(s) in the defined trigger and action, while *Where* specifies the involved location(s). *Who* and *Where* can be generic (e.g., *any friends, any closed space, the location of the trigger*) or specific (e.g., *my friend Mark,* or *my kitchen*).

IoT Ecosystem

The **IoT Ecosystem** block models IoT devices and services as entities that can offer one or more functionality. Each functionality may have commands to perform some actions, or notifications to register event listeners. Commands and notifications include the features needed to interact with the specific technology (e.g., specific commands with required parameters to be sent to devices and services). For example, a *Philips Hue* lamp is modeled as a *Lamp* (a particular type of *Lighting System*) that offers, among the others, an *Actuating* functionality with *Turn On* and *Turn Off* commands.

Contextual Information

The **Contextual Information** block describes locations and users that act as restrictions for trigger-action rules (i.e., *Who* and *Where* restrictions), and the contextual information of the IoT Ecosystem (e.g., the position of a device, or the users subscribed to an online platform). Thanks to this layer, trigger-action rules can be adapted at run-time to different users and (even unknown) locations. In this way, the ontological representation provides a strong support for executing the defined end-user rules.

Semantic Reasoning

The **Semantic Reasoning** block automatically maps the defined trigger-action rules with devices and services in the IoT Ecosystem that *allow to* reproduce the desired behaviors, by keeping into account the current context, dynamically. Semantic Reasoning is implemented with a set of built-in SWRL rules, that can be used from run-time environments for executing the trigger-action rules defined by end-users. For the trigger-action rule defined by Susan, for example, a specific SWRL rule maps all the *Lighting* devices of the user that can be turned on (i.e., the *Philips Hue* and *LIFX* lights, and the *Stack Lighting* lamp) as potentially able to reproduce the action behavior. By reasoning on the contextual information, e.g., by comparing the location of the lights with the locations of the user, run-time environments can easily select the right IoT object(s). It is worth noticing that abstract behaviors, e.g., *illuminate a place,* could be potentially reproduced in different ways according to the current context. Different methodologies can be adopted to select specific devices or services, ranging from preference-based approaches to fully-automatic solutions (e.g., through machine learning).

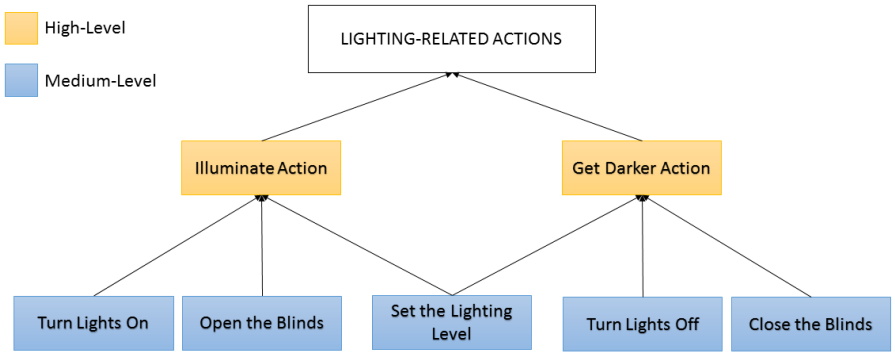

Figure 3: A Portion of the Lighting-related Actions in the EUPont Trigger-Action Programming Hierarchy.

Composing Trigger-Action Rules with EUPont

Since the main goal of EUPont is to help end-users in composing IoT applications, we evaluated this aspect in a previous work [2]. We were interested in the understandability of the model, and whether and how EUPont helped end-users create trigger-action rules. For this purpose, we compared the level of abstraction offered by EUPont with the representation adopted by IFTTT.

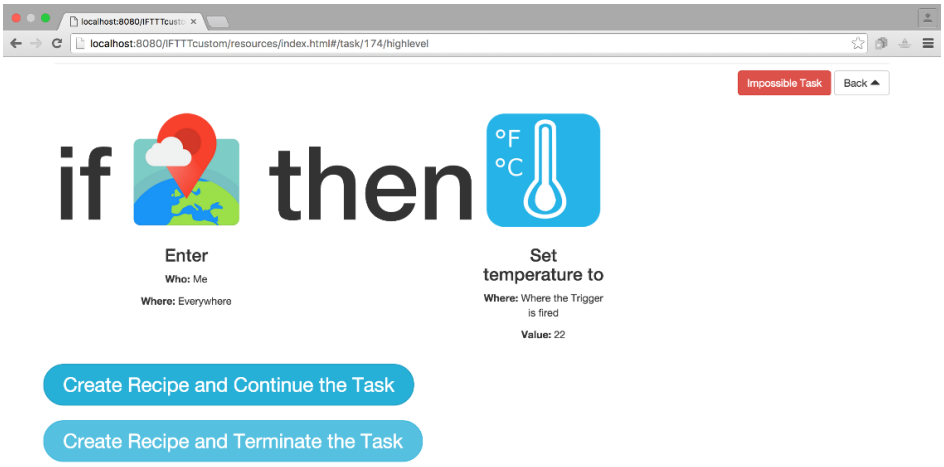

Figure 4: The Interface for Composing Trigger-Action Rules with EUPont

We created an interface modeled after IFTTT in two versions, one that allowed the composition of trigger-action rules in the EUPont representation (Figure 4), and the other that exploited a subset of the (low-level) devices, services, triggers, and actions as modeled by IFTTT. Then, we performed a user study with 10 participants, and asked them to complete five scenario-based tasks related to the creation of IoT applications with both versions of the interface.

The study we conducted revealed that EUPont is well understood by end users and suitable for creating IoT applications. Moreover, the high-level of abstraction of EUPont allows the creation of IoT applications in a more effective and efficient way than the representation currently adopted by IFTTT. Since EUPont abstracts and generalizes IoT devices and services, the number of rules created by the users to reach their goals was lower (28% fewer rules, on average). Additionally, the needed time and

the number of errors in composing trigger-action rules were significantly lower with EUPont than with the IFTTT-like representation. The needed time, in fact, was 52% lower with the High-Level model while the number of errors was 64% lower, in average. Users qualitatively reported appreciating the lower cognitive effort required during the composition of high-level rules and envisioned a potentiality for a greater discoverability and applicability of such rules.

EUPont Expressive Power

To use EUPont in practice, its expressiveness is very important. To improve the expressive power of the trigger-action approach adopted by contemporary solutions, EUPont already implements guidelines reported in the literature, e.g., multiple triggers and actions, trigger and action restrictions.
To further investigate the expressive power of EUPont and assess its completeness, we focused on the following two research questions:

- Is EUPont *at least as* expressive as the representations used by contemporary IoT end-user programming environments?
- Is EUPont *compatible* with low-level trigger-action rules defined with contemporary IoT end-user programming environments?

To answer these questions, we used a dataset of rules publicly shared on IFTTT [3] as of September 2016. First, we analyzed and pre-processed the dataset. Then, we translated all the trigger-action rules of the dataset in their corresponding Medium and High-Level rules, as modeled by EUPont. Two researchers were involved in this manual mapping. The resulting ontology is available at http://elite.polito.it/ontologies/eupont-ifttt.owl.

Translating Trigger-Action Rules in EUPont

Table 1 (column "Original dataset") reports some statistics about the original dataset. For each of the 295,156 rules, the dataset includes:

- id, creation date, description, and number of shares of the rule;
- trigger service, trigger name, and trigger description;
- action service, action name, and action description.

Before translating the trigger-action rules in the EUPont representation, we performed a data pre-processing step. Since we were interested in the final behaviors of the defined rules, we identified in the dataset the rules composed of ambiguous triggers or actions in terms of functionality.

For example, the rule "*if* the Wemo switch is turned on, *then* send me an Android SMS" has an ambiguous trigger, because we do not know which devices are connected to the switch. Another example is the rule "*if* the Nest cam detect a new motion, *then* execute a scene with my IntesisHome hub". In this case, the rule has an ambiguous action, because the scene that can be activated on the hub is defined by the user, and it can include many different environment settings that are not reported in the dataset.

For all the identified ambiguous rules, we manually inspected the description field, trying to discover more information about actual devices involved in triggers and actions and the user's intent. We deleted from the dataset 4,193 rules for which it was impossible to resolve the ambiguity. Finally, the pre-

processed dataset was composed of 290,963 rules composed by 127,173 different users (Table 1, column “After pre-processing").

	Original dataset	**After pre-processing**
Rules	295,156	290,964
Distinct Triggers	976	951
Distinct Actions	551	528
Users	129,206	127,173
Rules for each user (in average)	2.29 (S.D. = 6.5)	2.29 (S.D. = 6.5)
Max number of rules for a user	982	982
Min number of rules for a user	1	1

Table 1: Dataset Description

After data pre-processing, we carried out the translation process in three distinct phases.

1. **Triggers and Actions insertion.** In the first phase, we manually mapped the IFTTT triggers and actions found onto the dataset in the hierarchical organization of the Trigger-Action Programming block of EUPont.
2. **Rule insertion.** In the second phase, we automatically inserted the IFTTT rules of the dataset in the ontology by connecting each of them with its trigger and its action. For this purpose, we developed a Java program that uses the OWL API library [18] to work with the ontology.
3. **Rule translation.** After the insertion, the program retrieved the corresponding Medium Level Trigger and Action in the Trigger-Action Programming hierarchy for each trigger and action of the inserted IFTTT rules. Finally, for each user, the program removed the duplicated rules resulting after the Medium Level translation, and saved the results. The process was then repeated, separately, by considering the High-Level abstraction.

Results and Discussion

Table 2 reports the results of the translation process in both abstraction levels. All the 290,963 IFTTT rules in the pre-processed dataset have been translated. With respect to the original dataset, EUPont allowed to translate 98.58% of the rules, along with 97.44% of triggers and 95.83% of actions. Furthermore, as already explained, the 4,193 ambiguous rules are mainly due to a lack of information in the original dataset: by knowing the functionality defined by the users, also those rules could be translated.

	Medium-Level Translation	**High-Level Translation**
Number of translated rules	290,963	290,963
Number of rules after translation	179,110	170,353
Number of saved rules	111,853	120,610
% of saved rules	37.90%	41.45%
% of saved rules for each user	12.26% (S.D. = 24.12%)	13.26% (S.D. = 25.18%)

Table 2: Translation Results

The translation confirms that EUPont could significantly reduce the number of rules needed by end-users to satisfy their needs. With the Medium Level translation, the total number of rules could be reduced by 37.90%, and, in average, each user could save the 12.26% of their rules. By expressing the rules in an even more abstract way, i.e., with the High-Level of abstraction, the percentage of saved

rules increases (41.45% in total, and with an average saving of 13.26% for each user). Moreover, such promising results are influenced by the dataset distribution. In fact, 87,796 users (i.e., 68%) share one rule, only. In this case, obviously, the translation does not influence the final number of rules for the users but may avoid the creation of similar rules in the future.

Given the translation results, we can conclude that EUPont is as least as expressive than the representation model used by IFTTT, and it is fully compatible with low-level rules as defined in IFTTT. Moreover, the flexible trigger-action approach adopted by EUPont (e.g., with multiple triggers and actions) increases the expressiveness of the representation.

Conclusions and Future Works

The forthcoming IoT world presents its own set of issues, mainly in terms of interoperability between different technologies and brands. Such issues also influence the ways in which end-users can customize the joint behavior of their IoT devices and services.

In this paper, we described and evaluated EUPont, an ontology that allows end-users to define generic and technology-independent trigger-action rules that can be adapted to different contextual situations. We have shown that EUPont is more expressive than the most common solution in this field, i.e., IFTTT, because a) it follows literature guidelines to improve the expressive power of the trigger-action programming approach, and b) it is able to represent more than 200,000 IFTTT rules. EUPont allows to overcome the limitations of current low-level approaches, since IoT applications are more general and can adapt to different contextual situations as well to yet unknown IoT objects. Furthermore, EUPont allows end-users to define their needs with fewer rules. This aspect has a positive impact on the time needed by users to define their IoT applications, on the correctness level of the defined rules, and aims at increasing the discoverability and the reuse of such rules.

We believe that EUPont could serve as the core information layer for future IoT end-user programming solutions. Numerous challenges still remain: which level of abstraction offered by EUPont would end-users prefer, and how could a system choose which devices or services control in case of multiple possibilities? These and other questions are the focus of our current work.

Author Biographies

Fulvio Corno is an Associate Professor at the Department of Control and Computer Engineering of Politecnico di Torino since 2002. He is member of IEEE, IEEE Computer Society, and ACM. His research interests include Ambient Intelligence and the Internet of Things. He is also Associate Editor of the IT Professional IEEE Magazine. Contact him at fulvio.corno@polito.it

Luigi De Russis is a postdoc research assistant at the Department of Control and Computer Engineering of Politecnico di Torino. He received his Ph.D. in Computer Engineering from Politecnico di Torino, in 2014. He is a member of IEEE, IEEE Computer Society, and ACM. His research interests include Human-Computer Interaction and Ambient Intelligence. Contact him at luigi.derussis@polito.it

Alberto Monge Roffarello is a first-year Ph.D. student at the Department of Control and Computer Engineering of Politecnico di Torino. He received his M.S. in Computer Engineering from Politecnico di Torino, in 2015. His research interests span Human-Computer Interaction and Semantic Web. Contact him at alberto.monge@polito.it

References

[1] "IFTTT," [Online]. Available: https://ifttt.com. [Accessed 10 03 2017].

[2] F. Corno, L. De Russis and A. Monge Roffarello, "A High-Level Approach Towards End User Development in the IoT," in *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, Denver, CO (USA), 2017.

[3] B. Ur, M. P. Yong Ho, S. Brawner, J. Lee, S. Mennicken, N. Picard, D. Schulze and M. L. Littman, "Trigger-Action Programming in the Wild: An Analysis of 200,000 IFTTT Recipes," in *CHI 2016: The 34th Annual CHI Conference on Human Factors in Computing Systems*, San Jose, CA, (USA), 2016.

[4] H. Lieberman, F. Paternò, M. Klann and V. Wulf, "End-User Development: An Emerging Paradigm," in *End User Development*, 2006, pp. 1-8.

[5] A. K. Dey, T. Sohn, S. Streng and J. Kodama, "iCAP: Interactive Prototyping of Context-Aware Applications," in *Proceedings of the 4th International Conference on Pervasive Computing, PERVASIVE '06*, 2006.

[6] J. Lee, L. Garduño, E. Walker and W. Burleson, "A Tangible Programming Tool for Creation of Context-Aware Applications," in *Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, Ubicomp '13*, 2013.

[7] J. Danado and F. Paternò, "Puzzle: A mobile application development environment using a jigsaw methaphor," *Journal of Visual Languages and Computing,* vol. 25, no. 4, pp. 297-315, 2014.

[8] "Atooma," [Online]. Available: https://www.atooma.com/. [Accessed 10 03 2017].

[9] B. Ur, E. McManus, M. Pak Yong Ho and M. L. LIttman, "Practical Trigger-Action Programming in the Smart Home," in *Proceedings of the SIGCHI Conference of Human Factors in Computing Systems, CHI '14*, 2014.

[10] L. De Russis and F. Corno, "HomeRules: A Tangible End-User Programming Interface for Smart Homes," in *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, 2015.

[11] G. Ghiani, M. Manca and F. Paternò, "Authoring Context-dependent Cross-device User Interfaces Based on Trigger/Action Rules," in *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, 2015.

[12] "Ten Most Valued Internet of Things Startups," [Online]. Available: https://www.hottopics.ht/stories/lists/10-most-valued-internet-of-things-startups/. [Accessed 08 01 2016].

[13] "IFTTT Channels," [Online]. Available: https://ifttt.com/channels. [Accessed 08 01 2016].

[14] J. Huang and M. Cakmak, "Supporting Mental Model Accuracy in Trigger-action Programming," in *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, Osaka, Japan, 2015.

[15] B. R. Barricelli and S. Valtolina, "Designing for End-User Development in the Internet of Things," in *End-User Development*, vol. 9083, Springer International Publishing, 2015, pp. 9-24.

[16] G. Ghiani, M. Manca, F. Paternò and C. Santoro, "Personalization of Context-Dependent Applications Through Trigger-Action Rules," *ACM Transactions on Computer-Human Interaction,* vol. 24, no. 2, pp. 1-33, 2017.

[17] T. Berners-Lee, J. Hendler and O. Lassila, "The semantic web," *Scientific American,* vol. 284, no. 5, pp. 28-37, 2001.

[18] G. Desolda, C. Ardito and M. Matera, "Empowering End Users to Customize Their Smart Environments: Model, Composition Paradigms, and Domain-Specific Tools," *ACM Transactions on Computer-Human Interaction,* vol. 24, no. 2, pp. 12:1--12:52, 2017.

[19] "OWL API," [Online]. Available: http://owlapi.sourceforge.net. [Accessed 16 03 2017].